МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Видавничо-поліграфічний інститут
Кафедра репрографії

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт (комп'ютерного практикуму)
з дисципліни

«Технології створення Web-сторінок»

для напряму підготовки 6.051501

«Видавничо-поліграфічна справа»

спеціальності

«Технології електронних мультимедійних видань»

Київ – 2015

Методичні вказівки до виконання лабораторних робіт (комп'ютерного практикуму) з дисципліни «Технології створення Web-сторінок» для студентів напряму підготовки 6.051501 «Видавничо-поліграфічна справа» спеціальності «Технології електронних мультимедійних видань» для студ. Видавн.-полігр. ін.-ту / Укл. Б.Р. Кушлик, О.І. Кушлик-Дивульська − К.: НТУУ «КПІ». − 2015. − 58с.

Рекомендовано Вченою радою
Видавничо-поліграфічного інституту НТУУ «КПІ»
(Протокол № 2 від 28 вересня 2015 р.)

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт (комп'ютерного практикуму)
з дисципліни
«Технології створення Web-сторінок»

для напряму підготовки 6.051501
«Видавничо-поліграфічна справа»

спеціальності
«Технології електронних мультимедійних видань»

Укладачі: *Кушлик Богдан Ростиславович*
Кушлик-Дивульська Ольга Іванівна

Відповідальний редактор *О.М. Величко*, д-р техн. наук, проф.

Рецензенти: *О. В. Зоренко*, доцент, к.т.н.
Т. В. Розум, доцент, к.т.н.

ЗМІСТ

1. Мета, завдання і тематика лабораторних робіт (комп'ютерного практикуму)

Мета комп'ютерного практикуму полягає в закріпленні знань, одержаних студентами під час вивчення дисципліни «Технології створення Web-сторінок», їх застосуванні для вирішення конкретних завдань, сприянні самостійності у аналізі та прийнятті важливих професійних рішень, які є необхідною складовою підвищення технічного рівня підготовки студента. Зміст і структура методичних вказівок відображає новітні тенденції у питаннях створення та формування web-сторінок і забезпечує практичне вирішення завдань проектування складних за структурою, змістом та наповненням web-сторінок та їх запуск у реальне життя. Також метою лабораторних робіт (комп'ютерного практикуму) із даної дисципліни, яка є основоположною при формуванні майбутніх спеціалістів за спеціальністю «Технології електронних мультимедійних видань», є навчання майбутнього фахівця умінню вибрати й розрахувати найбільш ефективні моделі та засоби створення web-сторінок, підібрати до них відповідне устаткування та програмне забезпечення, проводити тестування робочих версій, виявляти помилки та усувати їх, забезпечувати запуск розроблених web-сторінок у професійне використання.

Лабораторні роботи (комп'ютерний практикум) виконуються в 6-му семестрі стаціонарної форми навчання за програмою підготовки бакалавр.

2. Основні вимоги до виконання лабораторних робіт (комп'ютерного практикуму)

Лабораторні роботи (комп'ютерний практикум) з дисципліни «Технології створення Web-сторінок» для кожного студента містять відповідні завдання. При виконанні лабораторних робіт (комп'ютерного практикуму) необхідно дотримуватися наведених нижче правил. Роботи, виконані без дотримання цих правил, можуть бути повернені студенту для доопрацювання. Протокол практичної роботи оформлюється у вигляді роздрукованих сторінок формату А4, оформлення якої здійснюється із дотриманням вимог ДСТУ.

Типова структура практичної роботи містить:

- титульний аркуш;
- аркуш завдання;
- основна частина;
- додатки (за необхідністю).

3. Зміст та перелік лабораторних робіт (комп'ютерного практикуму)

3.1. Лабораторна робота 1

Написання базового коду Web-сторінки у HTML5 та перевірка його працездатності.

Мета роботи: ознайомитись із базовими теговими командами, зрозуміти структуру типової web-сторінки, перевірити можливості роботи браузерів.

Хід роботи:

Лабораторна робота №1 складається з двох частин. У першій пропонується розглянути можливості HTML5 у порівнянні з HTML стосовно простоти написання базових команд. У другій частині пропонується розглянути можливості браузерів щодо їх підтримки розширених можливостей HTML5.

1. Написання базового коду

Надрукуйте вказаний нижче код із тегів у програмному продукті «Блокнот».

```
<!doctype html>
<html>
    <head>
        <title>My first web page</title>
        <meta charset=”utf-8”>
        <link rel=”stylesheet” href=”lounge.css”>
        <script src=”lounge.js”></script>
    </head>
<body>
    <h1> Вітаємо на дисципліні «Технології створення web-сторінок»</h1>
    <p>
        <img src=”drinks.gif” alt=”Drinks”>
```

Кожного вечора приєднуйтесь до роботи над створенням web-сторінок за чашкою улюбленого напою elixirs та можливо отримайте приз у грі «Отримай залік». Завжди до ваших послуг безпровідний інтернет, наданий ВПІ.

</body>

</html>

Перевірте працездатність коду та його сприйняття як web-сторінки, зберігши файл з розширенням *.html та відкривши його у будь-якому наявному браузері. Зробіть PrintScreen з відтвореної сторінки та додайте рисунок до звіту.

2. Перевірка можливостей браузерів.

Таблиця 1. Можливості браузерів щодо HTML5.

Браузер	Можливості						
	Video	Audio	Canvas	Web Storage	Geolocation	Web workers	Автономні веб-додатки
Firefox							
Safari							
Chrome							
Mobile Webkit							
Opera							
IE 6,7							
IE 8							
IE 9							

Вам доручено визначити поточний рівень підтримки по кожному з наведених нижче браузерів. Застосуйте пошук інформації через використання веб-посилання:

http://wickedlysmart.com/hfhtml5/browsersupport.html

Почергово відкрийте посилання у кожному із наявних на робочому комп’ютері браузерів та внесіть дані до таблиці 1.

Беріть до роботи лише найновіші версії браузерів. По кожному браузеру відзначте можливості, які й ньому підтримуються.

Оформлення звіту та порядок захисту

Лабораторна робота виконується на аркушах А4, в ній стисло відображається зміст, хід роботи та отримані результати. При захисті студент повинен розуміти зміст роботи та розуміти основне значення базових тегів, також знати відповіді на запитання щодо можливостей браузерів.

3.2. Лабораторна робота 2

Додавання JavaScript на сторінку у різних місцях. Перевірка працездатності коду.

Мета роботи: Зрозуміти мову JavaScript. Зрозуміти способи додавання кодів JavaScript до веб-сторінки. Зрозуміти спосіб розбирання браузером коду та шлях відтворення на сторінці.

Хід роботи:

Лабораторна робота №2 полягає у написанні коду, попередньо записаного на практичному занятті. Та додаванні його у базову веб-сторінку.

1. Наберіть код, повний текст якого наведено нижче:

```
<!doctype html>
<html>
    <head>
        <meta charset=”utf-8”>
        <title>My first JavaScript</title>
    </head>
    <body>
        <script>
Сюди необхідно вставити код JavaScript з практичного заняття «Написання циклів»
        </script>
    </body>
</html>
```

2. Додайте код з практичного заняття «Написання циклів» у вигляді окремого JavaScript-файлу, та додайте посилання на нього у коді основної сторінки.

3. Покажіть, що веб-сторінка відтворюється.

4. Змініть спосіб додавання Java-script до сторінки на інші два альтернативних та покажіть, що веб-сторінка коректно відображається.

Оформлення звіту та порядок захисту

Лабораторна робота виконується на аркушах А4, в ній стисло відображається зміст, хід роботи та отримані результати. У якості матеріалу, що повинен бути у звіті слід навести PrintScreen з робочої сторінки. При захисті студент повинен розуміти зміст роботи та розуміти основне значення основних команд блоків перехоплення та формування циклів.

3.3. Лабораторна робота 3

Випробування на практиці об'єктної моделі документа. Побудова сторінки за об'єктною моделлю документа.

Мета роботи: Розібрати спосіб створення об'єктної моделі документа за готовою веб-сторінкою. Створити веб-сторінку за наданою об'єктною моделлю документа.

Хід роботи:

1. Побудуйте об'єктну модель документа для коду наведеного нижче:

```
<!doctype html>
<html lang=”en”>
    <head>
        <title>Movies</title>
    </head>
    <body>
        <h1>Movie Showtimes</h1>
        <h2 id=”movie1”>Plan 9 from Outer Space</h2>
        <p>Playing at 3:00pm, 7:00pm.
            <span>
            Special showing tonight at <em>midnight</em>!
            </span>
        </p>
        <h2 id=”movie2”>Forbidden Planet</h2>
        <p>Playing at 5:00pm, 9:00pm.</p>
    </body>
</html>
```

2. Напишіть код для поданої нижче об'єктної моделі документа (рис.3.1.) та перевірте його працездатність, відкривши створену сторінку у браузері.

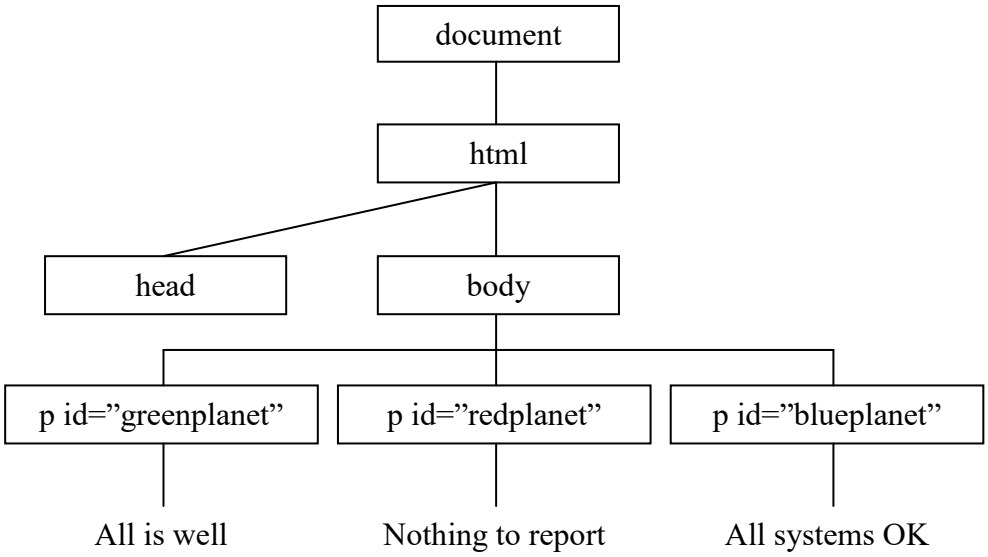

Рис. 3.1. Об’єктна модель документа.

3. У новоствореній сторінці додайте у якості дочірніх елементів до батьківського елементу <head> теги <title>, <meta> та <script>.

Всередині тегів розмістіть відповідну інформацію:

– У тегу <title> розмістіть назву

– У тегу <meta> вкажіть кодування

– У тегу <script> розмістіть команди, що дозволять робити заміну змісту, що знаходиться в елементі з ідентифікатором «greenplanet».

```
var planet = document.getElementByld("greenplanet");
planet.innerHTML = "Red Alert: hit by phaser fire!";
```

Перевірте код на працездатність, а саме чи змінився вміст елементу на сторінці з ідентифікатором «greenplanet».

4. Для коректної заміни вмісту тексту в елементі з ідентифікатором “greenplanet” необхідно, щоб JS-код виконувався строго після завантаження сторінки. Для досягнення цього необхідно застосувати JS-код команду

```
window.onload = init;
```

де init – це назва функції, яка буде викликатися після завантаження сторінки.

При цьому написаний раніше у тегу <script> JS-код необхідно помістити всередину функції init за прикладом:

```
function init() {
код
}
```

Перевірте працездатність коду після виконання всіх операцій.

Оформлення звіту та порядок захисту

Лабораторна робота виконується на аркушах А4, в ній стисло відображається зміст, хід роботи та отримані результати. У якості матеріалу, що повинен бути у звіті, слід навести рисунок об’єктної моделі документа та PrintScreen з робочої сторінки, написаної на основі заданої об’єктної моделі документа. При захисті студент повинен розуміти зміст роботи та розуміти способи додавання елементів до об’єктної моделі документа та відображення основних атрибутів об’єктів.

3.4. Лабораторна робота 4

Написання веб-додатку, що відображає список пісень у плейлисті за методом додавання їх із форми шляхом введення користувачем.

Мета роботи: Створити веб-додаток із застосуванням форм, з форматуванням списку та із записом нових елементів у об'єктну модель документа шляхом використання обробників подій.

Хід роботи:

1. Створити HTNL5-документ з формою та елементом списку, де буде міститися плейлист.

Введіть базовий код новоствореної веб-сторінки:

```
<!doctype html>
<html lang=”en”>
<head>
    <title>Webville Tunes</title>
    <meta charset=”utf-8”>
    <script scr=”playlist.js”></script>
    <link rel=”stylesheet” href=”playlist.css”>
</head>
<body>
    <form>
        <input type=”text” id=”songTextInput” size=”40” placeholder=”Song name”>
        <input type=”button” id=”addButton” value=”Add Song”>
    </form>
    <ul id=”playlist”>
    </ul>
</body>
</html>
```

2. Задання обробника події «click» для кнопки

Створимо функцію, яка буде викликатися при натисканні користувачем на кнопку «Add song» та перевіримо, що вона працює.

Задамо даний код у вигляді окремого JavaScript-файлу. Зверніть увагу, що першою командою повинна стати ініціалізація завантаження веб-сторінки.

```
window.onload = init;
function init() {
    var button = document.getElementById(“addButton”);
    button.onclick = handleButtonClick;
}
function handleButtonClick() {
    alert(“Button was clicked!”);
}
```

3. Написання обробника для витягування назви пісні із введеного користувачем тексту у форму.

```
function handleButtonClick() {
    var textInput = document.getElementById(“songTextInput”);
    var songName = textInput.value;
    if (songName == “”) {
        alert(“Please enter a song”);
    } else {
        alert(“Adding” + songName);
    }
}
```

4. Створення нового елементу об’єктної моделі документа для розміщення доданої користувачем пісні на веб-сторінку.

Зверніть увагу на наведену нижче об’єктну модель документа (рис. 3.2).

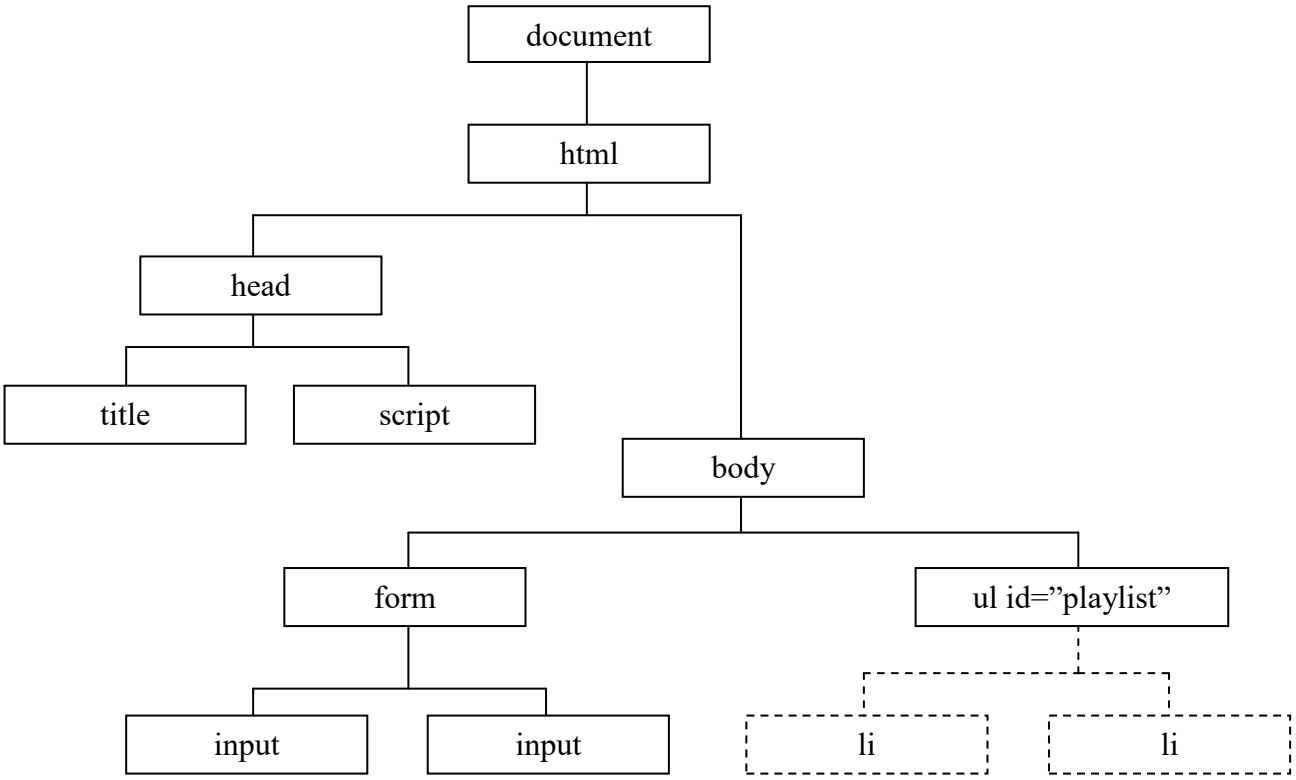

Рис. 3.2 Об’єктна модель документа до web-сторінки зі створення плейлиста.

У ній є елемент «ul id=”playlist”». Це і є власне необхідний нам плейлист. Зараз він пустий, його необхідно наповнити елементами, які буде додавати користувач.

Для цього необхідно ввести відповідний код:

```
var li = document.createElement(“li”);
li.innerHTML = songName;
```

5. Додавання новоствореного елементу у об’єктну модель документа сторінки.

```
function handleButtonClick() {
    var textInput = document.getElementById(“songTextInput”);
    var songName = textInput.value;
    var li = document.createElement(“li”);
    li.innerHTML = songName;
    var ul = document.getElementById(“playlist”)
```

```
        ul.appendChild(li);
}
```

Перевірте працездатність коду.

6. Станом на зараз створений автономно у веб-сторінці плейлист буде існувати лише поки відкрита веб-сторінка. При закритті він не буде зберігатись. Для того, щоб показати можливості зберігання та витягування даних скористаємось можливістю LocalStorage, яка є наявною у HTML5. Для цього необхідно набрати вказаний нижче код у окремий файл «playlist_store.js»:

```
function save(item) {
        var playlistArray = getStoreArray(“playlist”);
        playlistArray.push(item);
        localStorage.setItem(“playlist”, JSON.stringify(playlistArray));
}
function loadPlaylist() {
        var playlistArray = getSavedSongs();
        var ul = document.getElementById(“playlist”);
        if (playlistArray !=null) {
                for (var i=0; i<playlistArray.length; i++) {
                        var li = document.createElement(“li”);
                        li.innerHTML = playlistArray[i];
                        ul.appendChild(li);
                }
        }
}

function getSavedSongs() {
        return getStoreArray(“playlist”);
}
function getStoreArray(key) {
```

```
    var playlistArray = localStorage.getItem(key);
    if (playlistArray == null || playlistArray == “”) {
        playlistArray = new Array();
    }
    else {
        playlistArray = JSON.parse(playlistArray);
    }
    return playlistArray;
}
```

Також необхідно внести певні зміни у основний код робочої веб-сторінки.

Спершу додати посилання на playlist_store.js в елемент <head> у файлі playlist.html.

```
<script src=”playlist_store.js”></script>
<script src=”playlist.js”></script>
```

Також необхідно додати два рядки у вже існуючий код у файлі playlist.js, які будуть завантажувати та зберігати плейлист:

```
function init() {
    var button = document.getElementById(“addButton”);
    button.onclick = handleButtonClick;
    loadPlaylist();
}
function handleButtonClick() {
    var textInput = document.getElementById(“songTextInput”);
    var songName = textInput.value;
    var li = document.createElement(“li”);
    li.innerHTML = songName;
    var ul = document.getElementById(“playlist”)
    ul.appendChild(li);
    save(songName);
```

```
}
```

Оформлення звіту та порядок захисту

Лабораторна робота виконується на аркушах А4, в ній стисло відображається зміст, хід роботи та отримані результати.

У якості матеріалу, що повинен бути у звіті слід навести повний код робочої сторінки та файлів із кодом JavaScript.

При захисті студент повинен розуміти зміст роботи та розуміти спосіб написання обробника події та шлях, яким відбувається створення нових елементів та їх додавання до об’єктної моделі документа.

3.5. Лабораторна робота 5

Створення Web-додатку «Відтворення програми показу кінострічок» та його вдосконалення шляхом введення в нього конструкторів функцій та об'єктів із розширенням кількості відтворюваних об'єктів на Web-сторінці.

Мета роботи: Засвоїти навички з формування об'єктів у мові JavaScript. Зрозуміти принцип формування конструкторів об'єктів. Вдосконалити розуміння принципу побудови та логіки створення власних функцій.

Хід роботи:

1. Створіть базовий код HTML для сторінки:

```
<!doctype html>
<html lang=”en”>
    <head>
        <title>Webville cinema</title>
    </head>
    <body>
        <script src=”movie_showtimes.js”> </script>
    </body>
</html>
```

Також створіть js-файл, у який необхідно ввести наступну інформацію:

А) Дані про об'єкти (кінострічки з їх властивостями), з практичного заняття щодо створення об'єктів із Теми 4.

Б) Функцію, що викликатиме кожного разу нове діалогове вікно з повідомленням про показ чергового фільму залежно від часу доби.

```
function getNextShowing(movie) {
    var now = new Date().getTime();
    for (var i = 0; i < movie.showtimes.length; i++) {
        var showtime = getTimeFromString(movie.showtimes[i]);
        if ((showtime - now) > 0) {
```

```
                return "Next showing of " + movie.title + " is " +
                movie.showtimes[i];
                }
            }
        return null;
}
```

В) Функцію, що перетворює рядок формату, наприклад, 1am чи 6 pm (варто час показу у об’єктах з пункту А відобразити саме в такому вигляді) у значення часу в мілісекундах

```
function getTimeFromString(timeString) {
        var theTime = new Date();
        var time = timeString.match(/(\d+)(?::(\d\d))?\s*(p?)/);
        theTime.setHours( parseInt(time[1]) + (time[3] ? 12 : 0) );
        theTime.setMinutes( parseInt(time[2]) || 0 );
        return theTime.getTime();
}
```

Г) Команди, що викликатимуть черговий час показу фільму із об’єктів 1 та 2.

```
var nextShowing = getNextShowing(movie1);
alert(nextShowing);
nextShowing = getNextShowing(movie2);
alert(nextShowing);
```

Проведіть тест-драйв. Перевірте, чи з’являється діалогове віконце і що у ньому написано. Зробіть PrintScreen для протоколу лабораторної роботи.

2. Враховуючи Ваші знання з об’єктів, внесіть функцію виклику чергового показу getNextShowing як метод до об’єкту movie1.

```
var movie1 = {
        title: "Plan 9 from Outer Space",
        genre: "Cult Classic",
        rating: 5,
```

```
    showtimes: ["3:00pm", "7:00pm", "11:00pm"],
    getNextShowing: function(movie) {
      var now = new Date().getTime();
      for (var i = 0; i < movie.showtimes.length; i++) {
          var showtime = getTimeFromString(movie.showtimes[i]);
          if ((showtime - now) > 0) {
          return "Next showing of " + movie.title + " is " + movie.showtimes[i];
          }
      }
    return null;
    }
};
```

Оцініть можливість працездатності коду. Чи допустимий виклик функції шляхом, як ми робили це раніше

var nextShowing = getNextShowing(movie1);

за умови переміщення функції у метод об’єкту movie1?

Подумайте, що необхідно для того, щоб виклик функції відбувався таким чином: var nextShowing = movie1.getNextShowing();

Приберіть із функції getNextShowing, яка виступає методом об’єкту movie1 слово movie із усіх точечних нотацій. Та замініть його на ключове слово this, яке буде здійснювати ту саму роботу, однак стосовно конкретно даного об’єкту, в якому воно знаходиться.

Проведіть перевірку працездатності коду, однак змініть коди виклику команд наступним чином:

var nextShowing = movie1.getNextShowing();

alert(nextShowing);

nextShowing = movie2.getNextShowing();

alert(nextShowing);

Зробіть PrintScrenn для протоколу лабораторної роботи.

3. Створення конструктору об’єктів за прикладом раніше створених об’єктів movie1 та movie2.

```
function Movie(title, genre, rating, showtimes) {
    this.title = title;
    this.genre = genre;
    this.rating= rating;
    this.showtimes = showtimes;
    this.getNextShowing = function() {
        var now = new Date().getTime();
        for (var i = 0; i < this.showtimes.length; i++) {
            var showtime = getTimeFromString(this.showtimes[i]);
            if ((showtime - now) > 0) {
                return "Next showing of " + this.title+ " is " +
                this.showtimes[i];
            }
        }
    };
}
```

Дана функція дозволить створювати нові об’єкти, що матимуть вигляд за прикладом створених раніше вручну, однак виклик повідомлення про показ наступної кінострічки значно спроститься.

Видалимо весь попередній код із js-файлу, залишивши в ньому лише функцію Movie із цього пункту та функцію getTimeFromString із пункту 1В.

Додайте код для створення нових трьох об’єктів за прикладом:

```
var banzaiMovie = new Movie("Buckaroo Banzai",
                            "Cult Classic",
                            5,
                            ["1:00pm", "5:00pm", "7:00pm", "11:00pm"]);
var plan9Movie = new Movie("Plan 9 from Outer Space",
```

```
                                    "Cult Classic",
                                    2,
                                    ["3:00pm", "7:00pm", "11:00pm"]);
var forbiddenPlanetMovie = new Movie("Forbidden Planet",
                                    "Classic Sci-fi",
                                    5,
                                    ["5:00pm", "9:00pm"]);
```

Також раніше використовуваний код виклику діалогових повідомлень буде суттєво спрощений за прикладом:

```
alert(banzaiMovie.getNextShowing());
alert(plan9Movie.getNextShowing());
alert(forbiddenPlanetMovie.getNextShowing());
```

Проведіть перевірку працездатності коду із копією зображення з екрану до протоколу лабораторної роботи.

Оформлення звіту та порядок захисту

Лабораторна робота виконується на аркушах А4, в ній стисло відображається зміст, хід роботи та отримані результати.

У якості матеріалу, що повинен бути у звіті слід навести повний фінальний код робочої сторінки та файлів із кодом JavaScript, у якості доповнення обов’язково використовуйте копії зображення екрану відкритих діалогових повідомлень.

При захисті студент повинен розуміти зміст роботи та знати спосіб написання конструктора об’єктів.

3.6. Лабораторна робота 6

Записування координат в API-інтерфейсі Geolocation та представлення місця розташування користувача у використаному Web-додатку «Google Maps»: масштабування, центрування за координатами із заданою точністю. Написання коду для визначення місцезнаходження у часі. Побудова маршрутів на карті у режимі реального часу на створеній Web-сторінці.

Мета роботи: Створити веб-додаток із застосуванням API-інтерфейсу Geolocation для визначення поточного місця знаходження.

Хід роботи:

Частина 1. Представлення карти, визначення координат поточного місцезнаходження, розрахунок дистанції, відображення маркера на карті з інформаційним вікном.

1. Створіть базовий код сторінки та базовий JS-код у файлах myLoc.html та myLoc.js відповідно.

```
<!doctype html>
<html>
    <head>
        <meta charset = “utf – 8”
        <title>Where am I? </title>
        <script src ="myLoc.js"></script>
    </head>
    <body>
        <div id ="location">
        Your location will go here.
        </div>
    </body>
</html>
```

```
window.onload=getMyLocation;
function getMyLocation(){
        if (navigator.geolocation){
                navigator.geolocation.getCurrentPosition(displayLocation);
        }else{
                alert("Oops, no geolocation support");
        }
}
function displayLocation (position){
        var latitude = position.coords.latitude;
        var 1ongitude = position.coords.1ongitude;
        var div = document.getElementById("location");
        div.innerHTML = "You are at Latitude: " +latitude +" , Longitude: " +
longitude;
}
```

Проведіть перевірку працездатності написаного коду. Браузер у випадку підтримки ним API-інтерфейсу Geolocation повинен задати запитання чи надаєте ви йому доступ до визначення вашого поточного місцезнаходження. Після підтвердження на сторінці повинні бути відображені Ваші поточні координати, визначені за одним із способів, доступних браузеру (рис. 3.3)

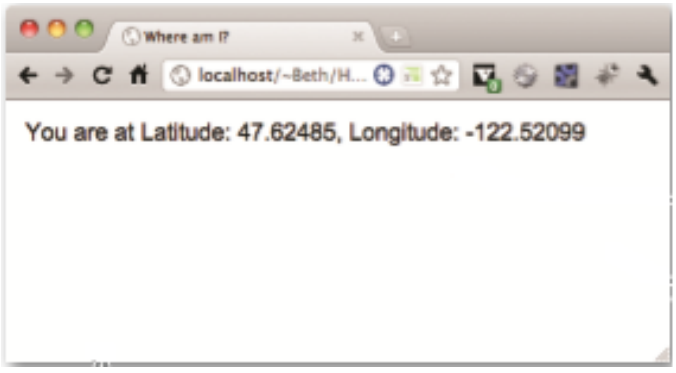

Рис. 3.3. Приклад отриманих координат.

2. Створіть обробник для помилки виконання дії з отримання даних щодо вашого поточного місцезнаходження.

Функція getcurrentPosition (successHandler, errorHandler, options) має у якості параметрів три варіанти, що вказуються строго у відповідному порядку як це показано вище. Ми у попередньому пункті записали successHandler (команду, точніше функцію, що виконується у випадку успіху), який назвали відповідною функцією displayLocation.

Необхідно аналогічним чином створити errorHandler дописавши відповідний код у файлі myLoc.js.

```
function displayError(error) {
    var errorTypes = {
        0: "Unknown error",
        1: "Permission denied by user",
        2: "Position is not available",
        3: "Request timed out"
    };
    var errorMessage = errorTypes[error.code];
    if (error.code == 0 || error.code == 2) {
        errorMessage = errorMessage + " " + error.message;
    }
    var div = document.getElementById("location");
    div.innerHTML = errorMessage;
}
```

Перевірте код на працездатність, викликавши одну або декілька можливих помилок.

3. Розрахунок дистанції між точками: вашим поточним місцезнаходженням та знаходженням штаб-квартири авторів сайту www.wickedlysmart.com.

Необхідно додати новий елемент у HTML-розмітку:

```
<body>
    <div id="location">
        Your location will go here.
    </div>
    <div id="distance">
        Distance from WickedlySmart HQ will go here.
    </div>
</body>
```

Необхідно також у js-файл додати функцію щодо вирахування координат за формулою гаверсинуса, реалізацію якої подано нижче:

```
function computeDistance(startCoords, destCoords) {
    var startLatRads = degreesToRadians(startCoords.latitude);
    var startLongRads = degreesToRadians(startCoords.longitude);
    var destLatRads = degreesToRadians(destCoords.latitude);
    var destLongRads = degreesToRadians(destCoords.longitude);
    var Radius = 6371; // вказаний радіус землі у км
    var distance = Math.acos(Math.sin(startLatRads) * Math.sin(destLatRads) +
                Math.cos(startLatRads) * Math.cos(destLatRads) *
                Math.cos(startLongRads - destLongRads)) * Radius;
    return distance;
}
function degreesToRadians(degrees) {
    var radians = (degrees * Math.PI)/180;
    return radians;
}
```

Необхідно також ввести координати із місцезнаходженням штаб-квартири авторів сайту www.wickedlysmart.com у той самий файл у якості глобальної змінної

```
var ourCoords =  {
    latitude: 47.624851,
```

longitude: -122.52099 };

Варто внести зміни до функції displayLocation із передаванням необхідних координат в ній для розрахунку щойно раніше створеній функції computeDistance.

```
function displayLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    var div = document.getElementById("location");
    div.innerHTML = "You are at Latitude: " + latitude + ", Longitude: " + longitude;
    var km = computeDistance(position.coords, ourCoords);
    var distance = document.getElementById("distance");
    distance.innerHTML = "You are " + km + " km from the WickedlySmart HQ"; }
```

Перевірте працездатність коду. Результат повинен бути на зразок представленого на рис. 3.4.

Рис.3.4. Приклад відтворення поточний координат та дистанції між вказаними точками.

Також перевірте свій результат на відповідність отриманому за посиланням: http://wickedlysmart.com/hfhtml5/chapter5/distance/myLoc.html

4. Додавання карти до сторінки.

Для додавання карти на сторінку спершу необхідно задіяти зовнішній API-інтерфейс розроблений Google, а саме зробити посилання на скрипт у коді сторінки як це показано нижче:

```
<script src="http://maps.google.com/maps/api/js?sensor=true"></script>
```

Після цього в коді сторінки необхідно в кінці створити ще один розділ :

```
<div id="map">
</div>
```

Далі у JS-файл необхідно додати наступне:

```
var map;
function showMap(coords) {
    var googleLatAndLong = new google.maps.LatLng(coords.latitude, coords.longitude);
    var mapOptions = {
        zoom: 10,
        center: googleLatandLong,
        mapTypeId: google.maps.MapTypeId.ROADMAP
        };
    var mapDiv = document.getElementById("map");
    map = new google.maps.Map(mapDiv, mapOptions);}
```

Також у кінці функції displayLocation(position) необхідно додати рядок для відображення карти

```
showMap(position.coords);
```

Проведіть перевірку працездатності коді. У Вас повинна відобразитись карта, що буде відцентрована за поточними координатами Вашого місцезнаходження.

5. Приколювання маркера (pin) на карту із позначенням інформації при кліку на нього.

У JS-файлі необхідно дописати ще одну функцію та власне створити сам маркер:

```
function addMarker(map, latlong, title, content) {
    var markerOptions = {
        position: latlong,
        map: map,
        title: title,
        clickable: true
    };
    var marker = new google.maps.Marker(markerOptions);}
```

Для створення інформаційного віконечка, що відкриватиметься при кліку на новостворений маркер необхідно доповнити функцію addMarker таким кодом:

```
var infoWindowOptions = {
    content: content,
    position: latlong};
var infoWindow = new google.maps.InfoWindow(infoWindowOptions);
google.maps.event.addListener(marker, "click", function() {
infoWindow.open(map);
});
```

Для коректності відображення залишилось лише додати до функції showMap виклик функції addMarker із передаванням їй відповідних аргументів title та content:

```
var title = "Your Location";
var content = "You are here: " + coords.latitude + ", " + coords.longitude;
addMarker(map, googleLatandLong, title, content);
```

Перевірте результат, він повинен виглядати наприклад як показано на рис. 3.5 або як вказано за посиланням:

http://wickedlysmart.com/hfhtml5/chapter5/marker/myLoc.html

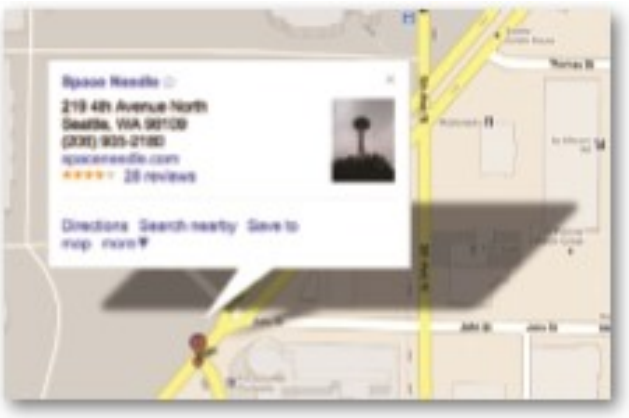

Рис. 3.5. Приклад кліку на маркер та появи інформаційного віконця.

Для додавання опції точності визначення вашого поточного місцезнаходження додайте наступний код до функції displayLocation:

```
function displayLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    var div = document.getElementById("location");
    div.innerHTML = "You are at Latitude: " + latitude + ", Longitude: " + longitude;
    div.innerHTML += " (with " + position.coords.accuracy + " meters accuracy)";
    var km = computeDistance(position.coords, ourCoords);
    var div = document.getElementById("distance");
    distance.innerHTML = "You are " + km + " km from the WickedlySmart HQ";
    showMap(position.coords);
}
```

Частина 2. Створення додатку для відслідковування координат поточного місцезнаходження у режимі реального часу.

1. Створення базової сторінки та доопрацювання базового js-коду.

Завданням є створити додаток, що мав би можливість при натисканні відповідної кнопки відслідковувати Ваше поточне місцезнаходження. Також повинна бути можливість припинити слідкування.

Для реалізації поставленого завдання використаємо код із першої частини даної лабораторної роботи, внісши у нього деякі зміни Зокрема додамо дві кнопки та змінимо назву:

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Wherever you go, there you are</title>
        <script src="myLoc.js"></script>
        <link rel="stylesheet" href="myLoc.css">
    </head>
    <body>
        <form>
            <input type="button" id="watch" value="Watch me">
            <input type="button" id="clearWatch" value="Clear watch">
        </form>
        <div id="location">Your location will go here.</div>
        <div id="distance">Distance from WickedlySmart HQ will go here.</div>
        <div id="map"></div>
    </body>
</html>
```

Також перепрацюємо старий код у функції getMyLocation:

```
function getMyLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(displayLocation,displayError);
```

```
        var watchButton = document.getElementById("watch");
        watchButton.onclick = watchLocation;
        var clearWatchButton = document.getElementById("clearWatch");
        clearWatchButton.onclick = clearWatch;
    }
    else {
        alert("Oops, no geolocation support");
    }
}
```

2. Написання обробників події click для новостворених кнопок watchButton та clearWatchButton.

Зверху js-файлу з основним кодом необхідно оголосити глобальну змінну:

```
var watchId = null;
```

Також в кінці слід дописати дві нові функції.

```
function watchLocation() {
    watchId = navigator.geolocation.watchPosition(displayLocation, displayError);
}
function clearWatch() {
        if (watchId) {
                navigator.geolocation.clearWatch(watchId);
                watchId = null;
        }
}
```

3. Необхідно внести невеликі зміни до функції displayLocation задля коректного відображення карти:

```
function displayLocation(position) {
        var latitude = position.coords.latitude;
        var longitude = position.coords.longitude;
        var div = document.getElementById("location");
```

```
    div.innerHTML = "You are at Latitude: " + latitude + ", Longitude: " +
longitude;
    div.innerHTML += " (with " + position.coords.accuracy + " meters
accuracy)";
    var km = computeDistance(position.coords, ourCoords);
    var distance = document.getElementById("distance");
    distance.innerHTML = "You are " + km + " km from the WickedlySmart
HQ";
    if (map == null) {
        showMap(position.coords);
    } else {
        scrollMapToPosition(position.coords);
    }
}
```

Перевірте працездатність коду завантаживши його на власний android чи IOS–планшет, можна на ноутбук чи на відповідний мобільний телефон. Увімкнуть кнопку «Watch position» та змініть своє місце розташування. За умови доступу до Інтернету на карті повинні з'являтися нові маркери із Вашим новим місцезнаходженням.

Також перевірити себе можна за зразком поданим за посиланням: http://wickedlysmart.com/hfhtml5/chapter5/watchme/myLoc.html.

Оформлення звіту та порядок захисту

Лабораторна робота виконується на аркушах А4, в ній стисло відображається зміст, хід роботи та отримані результати. У якості матеріалу, що повинен бути у звіті слід навести PrintScreen з робочої сторінки після кожного етапу описаного у ході роботи. При захисті студент повинен розуміти зміст роботи, команди, що вказані у коді, та розуміти способи додавання карти на сторінку та відображення позначки на ній.

3.7. Лабораторна робота 7

Створення автономного Web-додатку «Gumball sales» для запису кількості реалізацій товару із введенням в нього функцій запиту-відповіді Web-служб.

Мета роботи: Засвоїти навички зі здійснення запитів та отримання відповідей до віддалених веб-джерел певної протокольованої інформації. Здійснити запуск власного сервера для тестування додатку перш, ніж викладати його у мережу.

Хід роботи:

Частина 1. Застосування XMLHttpRequest щодо запитів.

1.1. Створіть базовий код HTML для сторінки:

```
<!doctype html>
<html lang="en">
    <head>
        <title>Mighty Gumball (JSON)</title>
        <meta charset="utf-8">
        <script src="mightygumball.js"></script>
        <link rel="stylesheet" href="mightygumball.css">
    </head>
    <body>
        <h1>Mighty Gumball Sales</h1>
        <div id="sales"></div>
    </body>
</html>
```

У файлі mightygumball.js напишіть обробник для методу window.onload

```
window.onload = function() {
    var url = "http://localhost/sales.json";
    var request = new xMLHttpRequest();
    request.open("GET", url);
    request.onload = function() {
```

```
        if (request.status == 200) {
            updateSales(request.responseText);
        }
    };
    request.send(null);
}
```

Допишіть функцію updateSales, щоб вона відображала результати отриманого запиту на сторінці.

```
function updateSales(responseText) {
    var salesDiv = document.getElementById("sales");
    salesDiv.innerHTML = responseText;
}
```

1.2. Встановіть власний веб-сервер на LocalHost для тестування працездатності коду перш, ніж розміщувати його на веб-ресурсі.

А) Якщо Ви використовуєте комп'ютер на базі ОС Windows, можете скористатись сервісом від Microsoft за посиланням:
http://www.microsoft.com/web/downloads/platform.aspx
або можете використати альтернативний варіант, наприклад:
http://www.wampserver.com/en/

Можете також використати будь-який із доступних та відкритих ресурсів щодо створення серверів на LocalHost.

Б) Якщо Ви використовуєте комп'ютер на базі MacOS, то в першу чергу слід пересвідчитись, чи включена у Вас опція WebSharing шляхом:
«символ яблука» > System Preferences > Sharing
та перевірте, чи стоїть відповідна галочка.

1.3. Створіть у папці з робочими файлами новий файл sales.json, куди вручну внесемо наступну інформацію.

```
[{"name":"ARTESIA","time":1308774240669,"sales":8},
{"name":"LOS ANGELES","time":1308774240669,"sales":2},
{"name":"PASADENA","time":1308774240669,"sales":8},
```

{"name":"STOCKTON","time":1308774240669,"sales":2},
{"name":"FRESNO","time":1308774240669,"sales":2},
{"name":"SPRING VALLEY","time":1308774240669,"sales":9},
{"name":"ELVERTA","time":1308774240669,"sales":5},
{"name":"SACRAMENTO","time":1308774240669,"sales":7},
{"name":"SAN MATEO","time":1308774240669,"sales":1}]

Та зробіть даний файл доступним з вашого локального сервера за прикладом: http://localhost/gumball/sales.json

Змініть також рядок з посиланням на ваш новостворений локальний сервер у функції-обробнику методу window.onload.

```
window.onload = function() {
    var url = "http://localhost/gumball/sales.json";
    var request = new XMLHttpRequest();
    request.open("GET", url);
    request.onload = function() {
        if (request.status == 200) {
            updateSales(request.responseText);
        }
    };
    request.send(null);
}
```

Впевніться що у робочій папці у Вас знаходяться повноцінні робочі файли HTML, JavaScript, JSON та за бажанням CSS і виконайте перевірку працездатності коду шляхом виклику Вашої сторінки через локальний сервер, наприклад:

http://localhost/gumball/mightygumball.html

Зробіть PrintScreen з робочої сторінки для вставлення рисунку до протоколу.

1.4. Зовнішній вигляд даної сторінки недосконалий, однак виконується важлива функція – отримуються дані з іншого файлу за запитом.

Однак нам необхідно зробити ще дещо, а саме покращити зовнішній вигляд отриманої сторінки, а також запустити її, отримуючи дані з реального сервера, що генерує дані відповідного до запиту формату.

Отже, спершу необхідно дані, отримані шляхом XMLHttpRequest перетворити у реальний JSON-об’єкт.

```
function updateSales(responseText) {
    var salesDiv = document.getElementById("sales");
    salesDiv.innerHTML = responseText;          //даний рядок не потрібен
    var sales = JSON.parse(responseText);
}
```

Далі, за допомогою циклу, передати нові об’єкти до DOM сторінки.

```
function updateSales(responseText) {
    var salesDiv = document.getElementById("sales");
    var sales = JSON.parse(responseText);
    for (var i = 0; i < sales.length; i++) {
        var sale = sales[i];
        var div = document.createElement("div");
        div.setattribute("class", "saleItem");
        div.innerHTML = sale.name + " sold " + sale.sales + " gumballs";
        salesDiv.appendChild(div);
    }
}
```

Перезавантажте сторінку, пересвідчіться, що код працює, та зробіть PrintScreen для вставлення зображення до протоколу.

1.5. Якщо тестування на локальному сервері пройшло успішно, змініть url-адресу на реальну:

```
window.onload = function() {
    var url = "http://gumball.wickedlysmart.com";
    var request = new xMLHttpRequest();
```

…

}

Перевірте отриманий результат. ***Код працює?***

Подумайте, над вирішенням питання, що саме необхідно зробити для вирішення завдання. Яким чином забезпечити уникнення браузерної політики безпеки із застосуванням XMLHttpRequest?

Варіанти своїх відповідей запишіть до протоколу.

Скористайтесь посиланням:
http://gumball.wickedlysmart.com/gumball/gumball.html
за яким всі файли були розміщені на умовному сервері замовника. Поясніть причину роботи коду в даному конкретному випадку.

Оформлення звіту та порядок захисту

Лабораторна робота виконується на аркушах А4, в ній стисло відображається зміст, хід роботи та отримані результати. У якості матеріалу, що повинен бути у звіті слід навести PrintScreen з робочої сторінки після кожного етапу описаного у ході роботи.

Також у протоколі повинен бути наведений повний код робочої сторінки, поетапні часткові варіанти наводити не потрібно.

При захисті студент повинен розуміти зміст роботи, команди, що вказані у коді, та розуміти шляхи виконання запитів, що використовуються веб-службами у HTML-5.

Лабораторна робота 8

Створення Web-додатку, що дозволяє створювати дизайн футболок використовуючи елементи Canvas та запиту-відповіді Web-служб.

Мета роботи: Засвоїти навички зі створення об’єктів Canvas для малювання на веб-сторінці. Створити автономний веб-додаток із застосуванням форм, вибору даних із форм та виконання обробки введених у форми даних у Canvas.

Хід роботи:

1. Створіть базовий код HTML для сторінки із введеним у нього елементом Canvas:

```
<!doctype html>
<html lang="en">
    <head>
        <title>Look What I Drew</title>
        <meta charset="utf-8">
        <style>
            canvas {
                border: 1px solid black;
            }
        </style>
    </head>
     <body>
        <canvas id="lookwhatIdrew" width="600" height="200"></canvas>
    </body>
</html>
```

2. Створіть форми, що дозволяють вибір:

– фону з альтернативами : білий та чорний;

– типу об’єктів, що з’являтимуться випадковим чином: круги чи квадрати;

– з вибором тексту кольору;

– з вибором джерела тексту (посилання на Tweet);

– кнопкою попереднього перегляду Preview;

– кнопкою замовлення футболки із отриманим зображенням (необхідний доступ до E-commerce на сервері, на якому буде розміщено готовий веб-додаток, тому у даній лабораторній роботі фактично робота цієї кнопки не є необхідною).

```
<form>
<p>
	<label for="backgroundColor">Select background color:</label>
	<select id="backgroundColor">
		<option value="white" selected="selected">White</option>
		<option value="black">Black</option>
	</select>
</p>
<p>
	<label for="shape">Circles or squares?</label>
	<select id="shape">
		<option value="none" selected="selected">Neither</option>
		<option value="circles">Circles</option>
		<option value="squares">Squares</option>
	</select>
</p>
<p>
	<label for="foregroundColor">Select text color:</label>
	<select id="foregroundColor">
		<option value="black" selected="selected">Black</option>
		<option value="white">White</option>
	</select>
</p>
<p>
	<label for="tweets">Pick a tweet:</label>
```

```
        <select id="tweets">
        </select>
</p>
<p>
        <input type="button" id="previewButton" value="Preview">
</p>
</form>
```

3. Внесіть посилання на JavaScript-файл, який буде містити обробники події клік на кнопку попереднього перегляду рисунку.

Також додайте повідомлення-помилку, що буде активним у разі, якщо браузер не підтримує API-інтерфейс Canvas.

```
<!doctype html>
<html lang="en">
        <head>
                <title>TweetShirt</title>
                <meta charset="utf-8" />
                <style>
                        canvas {
                                border: 1px solid black;
                        }
                </style>
                <script src="tweetshirt.js"></script>
        </head>
        <body>
                <h1>TweetShirt</h1>
                <canvas width="600" height="200" id="tshirtCanvas">
                <p>Please upgrade your browser to use TweetShirt!</p>
                </canvas>
                <form>
                // тут буде міститися код із формами
```

```
            </form>
        </body>
</html>
```

4. У файлі tweetshirt.js напишіть обробник для події click для кнопки Preview.

```
window.onload = function() {
        var button = document.getElementById("previewButton");
        button.onclick = previewHandler;
};
function previewHandler() {
        var canvas = document.getElementById("tshirtCanvas");
        var context = canvas.getContext("2d");
        var selectObj = document.getElementById("shape");
        var index = selectObj.selectedIndex;
        var shape = selectObj[index].value;
        if (shape == "squares") {
                for (var squares = 0; squares < 20; squares++) {
                        drawSquare(canvas, context);
                }
        }
}
```

Додайте функцію drawSquare для малювання довільно розташованих довільного розміру квадратів у заданих межах.

```
function drawSquare(canvas, context) {
        var w = Math.floor(Math.random() * 40);
        var x = Math.floor(Math.random() * canvas.width);
        var y = Math.floor(Math.random() * canvas.height);
        context.fillStyle = "lightblue";
        context.fillRect(x, y, w, w);
}
```

Перевірте працездатність коду шляхом відкривання сторінки, вибору відповідних опцій у формах (квадрати) та натисканням на кнопку Preview.

Квадрати з'явилися? Натисніть кнопку Preview декілька разів. Поясніть, чому нові квадрати з'являються, а старі не зникають? Як вирішити цю проблему?

5. Напишіть функцію fillBackgroundColor, яка заповнюватиме весь об'єкт Canvas фоном, який буде вказаний у формах вибору.

```
function fillBackgroundColor(canvas, context) {
    var selectObj = document.getElementById("backgroundColor");
    var index = selectObj.selectedIndex;
    var bgColor = selectObj.options[index].value;
    context.fillStyle = bgColor;
    context.fillRect(0, 0, canvas.width, canvas.height);
}
```

Додайте виклик функції fillBackgroundColor у функції-обробнику previewHandler так, щоб вона викликалась раніше рисування об'єктів вибору користувача (у нашому випадку поки-що тільки квадратів).

```
function previewHandler() {
    var canvas = document.getElementById("tshirtCanvas");
    var context = canvas.getContext("2d");
    fillBackgroundColor(canvas, context);
    var selectObj = document.getElementById("shape");
    var index = selectObj.selectedIndex;
    var shape = selectObj[index].value;
    if (shape == "squares") {
        for (var squares = 0; squares < 20; squares++) {
            drawSquare(canvas, context);
        }
    }
}
```

Перевірте виконання коду знову. Чи залишаються старі квадрати при повторному натисканні на кнопку Preview?

6. Допишіть код для рисування кругів, використовуючи знання, отримані на практичному занятті з даної теми щодо рисування ліній, секторів, кіл, та заповнення відповідних об’єктів.

```
function drawCircle(canvas, context) {
    var radius = Math.floor(Math.random() * 40);
    var x = Math.floor(Math.random() * canvas.width);
    var y = Math.floor(Math.random() * canvas.height);
    context.beginPath();
    context.arc(x, y, radius, 0, degreesToRadians(360), true);
    context.fillStyle = "lightblue";
    context.fill();
}
```

Додайте відповідний виклик функції drawCircle у функції previewHandler.

```
function previewHandler() {
    var canvas = document.getElementById("tshirtCanvas");
    var context = canvas.getContext("2d");
    fillBackgroundColor(canvas, context);
    var selectObj = document.getElementById("shape");
    var index = selectObj.selectedIndex;
    var shape = selectObj[index].value;
    if (shape == "squares") {
        for (var squares = 0; squares < 20; squares++) {
            drawSquare(canvas, context);
        }
    } else if (shape == "circles") {
        for (var circles = 0; circles < 20; circles++) {
            drawCircle(canvas, context);
```

```
            }
        }
}
```

7. Додавання способу витягування Твітів шляхом запиту у відповідної служби та розміщення тексту на полотні.

Спершу необхідно додати script з посиланням на TWITTER JSON API, який буде запитувати останній статус певного користувача.

```
<html>
...
        <body>
            <form>  ...  </form>
            <script
        src="http://twitter.com/statuses/user_timeline/wickedsmartly.json?
        callback=updateTweets">
            </script>
        </body>
</html>
```

Допишіть внизу файлу tweetshirt.js функцію, що буде надавати доступ до твітів для вибору їх із форми користувача.

```
function updateTweets(tweets) {
        var tweetsSelection = document.getElementById("tweets");
        for (var i = 0; i < tweets.length; i++) {
            tweet = tweets[i];
            var option = document.createElement("option");
            option.text = tweet.text;
            option.value = tweet.text.replace("\"", "'");
            tweetsSelection.options.add(option);
        }
        tweetsSelection.selectedIndex = 0;
}
```

Напишіть функцію drawText для рисування тексту, яка розташовуватиме стандартний текст зверху зліва та знизу справа, а також яка розташовуватиме обраний твіт по центру із іншими параметрами шрифтового оформлення.

```
function drawText(canvas, context) {
    var selectObj = document.getElementById("foregroundColor");
    var index = selectObj.selectedIndex;
    var fgColor = selectObj[index].value;

    context.fillStyle = fgColor;
    context.font = "bold 1em sans-serif";
    context.textalign = "left";
    context.fillText("I saw this tweet", 20, 40);

    selectObj = document.getElementById("tweets");
    index = selectObj.selectedIndex;
    var tweet = selectObj[index].value;
    context.font = "italic 1.2em serif";
    context.fillText(tweet, 30, 100);

    context.font = "bold 1em sans-serif";
    context.textalign = "right";
    context.fillText("and all I got was this lousy t-shirt!", canvas.width-20, canvas.height-40);
}
```

Проведіть перевірку працездатності коду.

8. Додавання рисунку твіттер-пташки до полотна Canvas. Збереження рисунку, намальованого у полотні Canvas для подальшого використання у різноманітному програмному забезпеченні.

Запитайте у викладача файл із зображенням твіттер-пташки і він вам його надасть. Розмістіть файл twitterBird.png у основну папку та опишіть функцію drawBird.

```
function drawBird(canvas, context) {
    var twitterBird = new Image();
    twitterBird.src = “twitterBird.png”;
    twitterBird.onload = function() {
        context.drawImage(twitterBird, 20, 120, 70, 70);
    };
}
```

Для збереження зображення нарисованого у Canvas необхідно створити функцію makeImage, яка при події click на елементі Canvas записуватиме файл у png-файл.

```
function makeImage() {
    var canvas = document.getElementById("tshirtCanvas");
    canvas.onclick = function () {
        window.location = canvas.toDataURL("image/png");
    };
}
```

Також необхідно дописати виклик даної функції у відповідному місці.

```
window.onload = function() {
    var button = document.getElementById("previewButton");
    button.onclick = previewHandler;
    makeImage();
```

}

Оформлення звіту та порядок захисту

Лабораторна робота виконується на аркушах А4, в ній стисло відображається зміст, хід роботи та отримані результати. У якості матеріалу, що повинен бути у звіті слід навести PrintScreen з робочої сторінки після кожного етапу описаного у ході роботи.

У протоколі повинен бути наведений повний код робочої сторінки, поетапні часткові варіанти наводити не потрібно.

При захисті студент повинен розуміти зміст роботи, команди, що вказані у коді, та розуміти способи рисування окремих об’єктів у Canvas.

Лабораторна робота 9

Задавання обробників завершення відео та функцій управління відео до створеної раніше Web-сторінки з відеоматеріалом. Додавання функції оброблення відео із застосуванням тимчасового буферу. Реалізація тимчасового буферу у Canvas.

Мета роботи: Засвоїти навички зі створення об’єктів аудіо та відео шляхом включення їх до веб-сторінки. Створити автономний веб-додаток із застосуванням форм, вибору даних із форм та виконання обробки введених у форми даних у Canvas.

Хід роботи:

1. Створіть базовий код HTML для сторінки із введеним у нього елементом відео.

```
<!doctype html>
 <html lang="en">
<head>
        <title>Webville Tv</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="webvilletv.css">
</head>
<body>
        <div id="tv">
                <div id="tvConsole">
                        <div id="highlight">
                                <img src="images/highlight.png" alt="highlight for tv">
                        </div>
                        <div id="videoDiv">
                                <video controls autoplay src="video/preroll.mp4"
                                width="480" height="360"
                                poster="images/prerollposter.jpg" id="video">
```

```
                    </video>
                </div>
            </div>
        </div>
</body>
</html>
```

Зверніть увагу, що Вами первинно був казаний лише один тип формату файлу для зчитування. <video controls autoplay src="video/preroll.mp4"
Даний тип прекрасно підходить для такого браузера як Safari, однак він не є доречним до застосування у браузерах Google Chome, Opera чи Mozilla Firefox, які використовують інші формати відео-файлів.

Отже, для того, щоб за замовчуванням, незалежно від використовуваного користувачем типу браузера, відео-файл відтворювався належним чином необхідно вписати таку частину коду до HTML-розмітки.

```
<video src="video/preroll.mp4" id="video"
        poster="video/prerollposter.jpg" controls
        width="480" height="360">
    <source src="video/preroll.mp4">
    <source src="video/preroll.webm">
    <source src="video/preroll.ogv">
<p>Sorry, your browser doesn’t support the video element</p>
</video>
```

Для вдосконалення системи підбирання форматів, враховуючи різноманітні кодеки для максимально якісного відтворення відео із різноманітними налаштуваннями, необхідно додати наступні атрибути:

```
<video id="video" poster="video/prerollposter.jpg" controls width="480" height="360">
```

<source src="video/preroll.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
<source src="video/preroll.webm" type='video/webm; codecs="vp8, vorbis"'>
<source src="video/preroll.ogv" type='video/ogg; codecs="theora, vorbis"'>
Sorry, your browser doesn't support the video element
</video>

Перевірте працездатність коду. Зверніть увагу, що у даному випадку використовується така опція як autoplay, що означає, що відео файл буде запускатись у момент відкривання веб-сторінки.

2. Приберемо опцію автоматичного запуску відтворення відео, та додамо опції ручного контролю за відео файлом шляхом написання відповідного webvilletv.js файлу, одночасно із додаванням списку програвання відео-файлів.

Додайте посилання на script у тегу <head>:

```
<script src="webvilletv.js"></script>
```

У самому файлі webvilletv.js необхідно записати наступний код:

```
var position = 0;
var playlist;
var video;
window.onload = function() {
    playlist = ["video/preroll.mp4",
        "video/areyoupopular.mp4",
        "video/destinationearth.mp4"];
    video = document.getElementById("video");
    video.addEventListener("ended", nextvideo, false);
    video.src = playlist[position];
    video.load();
    video.play();
}
```

```
function nextvideo() {
	position++;
	if (position >= playlist.length) {
		position = 0;
	}
	video.src = playlist[position];
	video.load();
	video.play();
}
```

Перевірте працездатність коду. Впевніться, що відео файли відтворюються коректно.

Оформлення звіту та порядок захисту

Лабораторна робота виконується на аркушах А4, в ній стисло відображається зміст, хід роботи та отримані результати. У якості матеріалу, що повинен бути у звіті слід навести PrintScreen з робочої сторінки після кожного етапу описаного у ході роботи.

У протоколі повинен бути наведений повний код робочої сторінки, поетапні часткові варіанти наводити не потрібно.

При захисті студент повинен розуміти зміст роботи та значення окремих команд коду.

4. Порядок захисту лабораторних робіт (комп’ютерного практикуму)

Лабораторна робота (комп’ютерний практикум) є самостійною роботою студента. Вона виконується на лабораторних заняттях і у час, відведений для самостійних занять. Лабораторні роботи виконуються у відповідності до графіку навчального процесу. Студент повинен пред’являти викладачу результати своєї роботи на лабораторних чи практичних заняттях. Захист лабораторних робіт відбувається прилюдно шляхом відповіді на поставлені викладачем запитання, стосовно теми лабораторної роботи.

Рекомендована література

1. Фримен Э., Робсон Э. Изучаем программирование на HTML5. – СПб.: Питер, 2013. – 640 с.
2. Фримен Э., Фримен Э. Изучаем HTML, XHTML и CSS. – СПб.: Питер, 2012. – 656 с.
3. Моррисон М. Изучаем JavaScript. – СПб.: Питер, 2013. – 592 с.
4. Фримен Э., Фримен Э., Сьерра К., Бейтс. Б. Паттерны проектирования. – СПб.: Питер, 2012. – 656 с.
5. Пилон Д. Пилон Т. Программируем для iPhone и iPad. – СПб.: Питер, 2013. – 592 с.
6. Рилон Д. Майлз Р. Управление разработкой ПО. – СПб.: Питер, 2013. – 464 с.
7. http://www.w3schools.com/
8. http://htmlbook.ru/
9. Стивен Хольцнер HTML5 за 10 минут [5-е издание] [Пер. с англ.] / М.: Издательский дом «Вильямс», 2011. – 240 с.
10. https://developer.mozilla.org/

Додаток А

Приклад оформлення титульного аркуша

МІНІСТРЕСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Видавничо-поліграфічний інститут

Кафедра репрографії

Лабораторна робота №1
З дисципліни «Технології створення Web-сторінок»
на тему «_____»

Виконав студент групи _____
_____

Перевірив
_____

КИЇВ – 201_